Задача

Реализовать отчёт, показывающий рост LTV дневных когорт игроков по дням жизни (Описание отчёта).

1. Используется база **PostgreSQL**, таблицы:
    - **player** - данные игроков
    - **payment** – данные платежей
2. Доступ к тестовой базе:

postgresql://observer:koh7theozoh8ohCh@example.c3eqbflfu0cv.eu-central-1.rds.amazonaws.com:5532/example

3. Нужно сделать: скрипт на Python, который подключается к базе, выполняет SQL-запрос и затем завершает обработку полученных данных. Итоговая таблица сохраняется в файл.
4. Опционально провести Дополнительные манипуляции.
5. Результат в виде архива (скрипт + инструкция по запуску) ждём в ответ на письмо с заданием.

Описание отчёта

UA менеджер хочет понимать, насколько хорошо платят игроки, пришедшие в игру в определённые даты.

Он хочет вставить в запрос **диапазон дат регистрации,** и получить на выходе таблицу такого вида:

Date	Installs	LTV	LTV-1	LTV-2	LTV-3	LTV-4
1 мая	1000	€ 0.4	€ 0.1	€ 0.2	€ 0.3	€ 0.4
2 мая	1200	€ 0.02	€ 0.01	€ 0.01	€ 0.02	
3 мая	800	€ 1.0	€ 0.8	€ 1.0		

(Допустим, сегодня 5-е мая, и он ввёл диапазон с 1 по 3 мая)

Таблица показывает рост **LTV** (LifeTime Value) игроков, зарегистрировавшихся в конкретный день (когорта) – то есть, сколько игра, в среднем, заработала с каждой установки в этот день, и как это значение меняется с течением времени. Это позволяет оценить, насколько успешно игроки конвертируются в платящих – например, в данном случае 2-го мая что-то явно пошло не так (маленький LTV и медленно растёт), зато 3-го мая было какое-то успешное решение (высокий LTV и быстро растёт).

- **Date** – Дата регистрации когорты;
- **Installs** – Количество установок игры в этот день;
- **LTV** – Итоговый LTV когорты;
- **LTV-1, 2, 3...** – LTV первого и последующих дней жизни.

Как считать LTV

LTV когороты – это сумма всех платежей когорты за всё время, поделённая на количество Installs когорты.
LTV дня жизни – это накопительная метрика. Сумма платежей за текущий и все предыдущие дни жизни, поделенная на количество Installs когорты. Она всегда только растёт со временем (либо не изменяется, если не было покупок).

Дополнительные данные

- UA менеджер самостоятельно вставит диапазон дат в запрос, согласно составленной вами инструкции;
- В образце представлены данные за 3 месяца, но отчёт должен учитывать боевые условия: UA менеджер может задать **любой диапазон дат**, в том числе и пол-года или год;
- Даты – **календарные**. То есть, неважно, в какое время дня зарегистрировался игрок. Он мог начать играть в 00:01 или в 23:59 1-го мая – и в обоих случаях он попадает в когорту 1-го мая. Его 1-й день жизни будет 1-го мая (даже если это была всего минута), а 2-го мая наступит 2-й день жизни;
- **Текущая дата** не учитывается в отчёте, так как день ещё не завершился.
- Обратите внимание, что мы считаем **LTV всей когорты**. То есть, сумму покупок нужно делить на **все инсталлы**, а не только на тех, кто заплатил.

Опционально

Основная сложность отчёта – корректно получить и расчитать данные о росте LTV, и правильно отобразить таблицу.
Тем не менее, отчёт можно сделать точнее за счёт использования дополнительной информации из базы данных.

Варианты

1. Сделать две таблицы: с **комиссией (revenue)** и без **комиссии (gross)** платформы;
2. Учесть, что некоторые платежи могли быть **отменены** игроком – при этом сумма платежа возвращается игроку;
3. Учесть, что новый игровой день наступает по GMT -1. В этот момент обновляются все ежедневные задачи игроков, и желательно статистику также считать в этой временной зоне. Все timestamp в базе данных в UTC;
4. Учесть, что платёж может быть тестовым.

Требования

1. Результаты скрипта должны совпасть с нашими;
2. Скрипт на Python и SQL-запрос должны быть оптимизированы на скорость исполнения;
3. Мы также будем учитывать организацию кода, пояснения и комментарии.